A PLATFORM FOR SUPERTREE ALGORITHMS

P Daniel and C Semple

Department of Mathematics and Statistics
University of Canterbury
Private Bag 4800
Christchurch, New Zealand

Report Number: UCDMS2004/4 MARCH 2004

A PLATFORM FOR SUPERTREE ALGORITHMS

PHILIP DANIEL AND CHARLES SEMPLE

ABSTRACT. Amalgamating smaller evolutionary trees into a single parent tree is an important task in evolutionary biology. Traditionally, the (supertree) methods used for this amalgamation take a collection of leaf-labelled trees as their input. However, it has been recently highlighted that, in practice, such an input is somewhat limiting and that one would like supertree methods for collections of trees in which some of the interior vertices, as well as all of the leaves, are labelled [10]. In this paper, we describe what appears to be the first general approach for constructing such methods and show that any method using this approach satisfies particular desirable properties.

1. INTRODUCTION

In evolutionary biology, *supertree methods* have become a fundamental process for constructing an evolutionary tree that best represents the information exhibited by the original input. These methods amalgamate an input collection of smaller evolutionary trees on overlapping sets of taxa into a single parent tree called a *supertree*. The increasing popularity of supertree methods is highlighted by a recent survey [3] and a soon-to-be published book [4].

If the input collection of trees carries no conflicting information, then one would like the resulting supertree to preserve all of the relationships displayed by each of the trees in this collection. For collections of rooted phylogenetic trees, there is a polynomial-time algorithm that finds such a tree. In practice, however, incompatibility is more common and so one seeks a method that resolves these conflicts in a sensible way, while still producing a supertree that has a number of attractive properties. The following list of desirable properties for any supertree method applied to a collection $\mathcal{P}$ of rooted phylogenetic trees is given in [12]:

(i) The method runs in polynomial time in the size of the input.
(ii) The resulting supertree displays all rooted binary subtrees shared by all of the trees in $\mathcal{P}$.
(iii) If $\mathcal{P}$ is compatible, then the resulting supertree displays each of the trees in $\mathcal{P}$.

Date: 26 February 2004.
1991 *Mathematics Subject Classification.* 05C05; 92D15.
Key words and phrases. Rooted phylogenetic tree; Nested taxa; Supertree.
The first author was supported by the New Zealand Institute of Mathematics and its Applications funded programme *Phylogenetic Genomics* and the second author was supported by the New Zealand Marsden Fund (UOC310).

(iv) The method satisfies the following two natural symmetry properties of *ordering* and *renaming*:
  (a) The resulting supertree is independent of the order in which the members of $\mathcal{P}$ are listed.
  (b) If we rename all the species, and then apply the method to this new collection of input trees, the resulting supertree tree is the one obtained by applying the method to the original collection $\mathcal{P}$, but with the species renamed as before.

(v) The method allows a possible weighting of the trees in $\mathcal{P}$.

To date, the algorithms MINCUTSUPERTREE [12] and its modified version [9] are the only two supertree methods for rooted phylogenetic trees that have been shown to satisfy all of the above properties. We remark here that (iv) may seem trivial to satisfy but, for collections of unrooted phylogenetic trees, it has been shown that no supertree method for such collections can simultaneously satisfy (iii) and both parts of (iv) [14].

In this paper, we present a platform for constructing supertree methods for collections of rooted semi-labelled trees; that is, rooted trees in which some (possibly none) of the interior vertices as well as all of the leaves are labelled. Making the extension from rooted phylogenetic trees to rooted semi-labelled trees means that we allow nested taxa in the input. In particular, the interior labels represent taxa at a higher taxonomic level than any of their descendants. For example, families versus genera or genera versus species. The main feature of this platform is that, provided the input satisfies two natural ancestor-descendant pairwise properties, any algorithm constructed from it satisfies all of the rooted semi-labelled tree analogues of the desirable properties (i)—(iii) above. Moreover, although the rooted semi-labelled tree analogues of (iv) and (v) are dependent on the constructed algorithm, satisfying these additional properties is not difficult. We highlight this with an example of such an algorithm. To the best of our knowledge, this is the first time such supertree methods for rooted semi-labelled trees have been considered. The next section contains some further background and necessary preliminaries for the rest of the paper.

2. BACKGROUND AND PRELIMINARIES

Throughout this paper, we will assume that the reader has some familiarity with the basics of phylogenetics. Unless otherwise stated, the notation and terminology follows Semple and Steel [13].

A *rooted phylogenetic tree (on X)* is an ordered pair $(T;\phi)$ consisting of a rooted tree T in which all interior vertices have degree at least three except the root which has degree at least two and a bijective map ϕ from X to the leaf set of T. Rooted phylogenetic trees on X are also called *rooted phylogenetic X-trees.*

Let T' be a rooted phylogenetic tree on X', and let X be a subset of X'. The *restriction* of T' to X is the rooted phylogenetic tree that is obtained from the minimal rooted subtree of T' induced by the elements of X by suppressing all

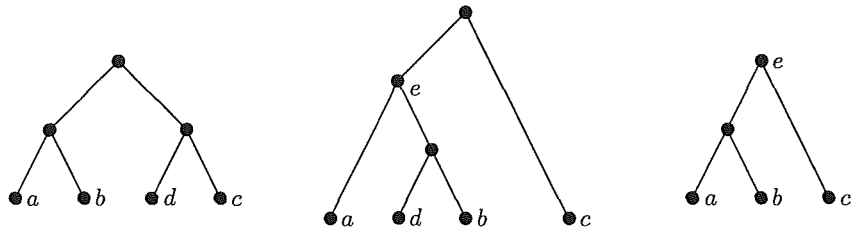


FIGURE 1. A collection $\mathcal{P}$ of rooted semi-labelled trees.

non-root vertices of degree two. This restriction is denoted by $\mathcal{T}'|X$. We say that $\mathcal{T}'$ *displays* a rooted phylogenetic X-tree $\mathcal{T}$ if, up to isomorphism, $\mathcal{T}'|X$ is a refinement of $\mathcal{T}$. Furthermore, a collection $\mathcal{P}$ of rooted phylogenetic trees is said to be *compatible* if there exist a rooted phylogenetic tree that displays each of the trees in $\mathcal{P}$. Intuitively, $\mathcal{P}$ is compatible if there is a rooted phylogenetic tree that preserves all of the ancestral relationships described by the trees in $\mathcal{P}$.

Traditionally, supertree methods have been applied to rooted phylogenetic trees. One of the first such methods is BUILD [2]. This polynomial-time algorithm takes a collection of rooted phylogenetic trees and determines if they are compatible, in which case it outputs a tree that displays each of the trees in the collection. Algorithms like BUILD are all-or-nothing algorithms as they only return a tree if they meet some criteria. However, despite this limitation, such algorithms give valuable insight to more general supertree methods. Indeed, the algorithm MINCUTSUPERTREE and its modified version is based on BUILD.

For nested taxa, the analogues of rooted phylogenetic trees and compatibility are 'rooted semi-labelled trees' and 'ancestral compatibility'. A *rooted semi-labelled tree (on X)* is an ordered pair $(T;\phi)$ consisting of a rooted tree T with vertex set V and root vertex ρ, and a map $\phi : X \to V$ with the properties that, for all $v \in V - \{\rho\}$ of degree at most two, $v \in \phi(X)$ and, if ρ has degree zero or one, then $\rho \in \phi(X)$. Rooted semi-labelled trees on X are also called *rooted X-trees*. Furthermore, if ϕ is one-to-one, then $(T;\phi)$ is said to be singularly labelled. Examples of rooted semi-labelled trees that are singularly labelled are shown in Fig. 1.

Let $X \subseteq X'$ and let $a, b \in X$. A rooted X'-tree $\mathcal{T}'$ *ancestrally displays* a rooted X-tree $\mathcal{T}$ if $\mathcal{T}'|X$ refines $\mathcal{T}$ so that, whenever a is a 'strict descendant' of b in $\mathcal{T}$, a is a 'strict descendant' of b in $\mathcal{T}'|X$. The formal definition of 'strict descendant' is given at the end of this section, but intuition should suffice for the moment. A collection $\mathcal{P}$ of rooted semi-labelled trees is *ancestrally compatible* if there is a rooted semi-labelled tree $\mathcal{T}$ that ancestrally displays each of the trees in $\mathcal{P}$, in which case we say that $\mathcal{T}$ *ancestrally displays* $\mathcal{P}$. Observe that if $\mathcal{P}$ consists of rooted phylogenetic trees, then $\mathcal{P}$ is compatible if and only if $\mathcal{P}$ is ancestrally compatible.

Page [10] recently motivated the problem of developing supertree methods for nested taxa and initially posed the problem of constructing a polynomial-time algorithm for determining the ancestral compatibility of an arbitrary collection of rooted semi-labelled trees. In answer to this problem, Daniel and Semple [7] presented an

algorithm called AncestralBuild. Analogous to Build, this polynomial-time algorithm is an all-or-nothing algorithm and determines if a collection $\mathcal{P}$ of rooted semi-labelled trees are ancestrally compatible, in which case it outputs a rooted semi-labelled tree that ancestrally displays $\mathcal{P}$. With AncestralBuild in hand, the next natural step forward is to construct a more general supertree method for rooted semi-labelled trees.

In Section 3 of this paper, we present a supertree method for collections $\mathcal{P}$ of rooted semi-labelled trees that are singularly labelled. Called NestedSupertree, this method either outputs a rooted semi-labelled tree, or a statement indicating that either there is a pair of taxa that are not 'pairwise consistent' or there is an 'ancestor-descendant contradiction'. Strictly speaking, this is still an all-or-nothing algorithm. However, such an inconsistency or a contradiction is very particular, and one that we believe in practice could be resolved separately. Based on AncestralBuild, one of the attractions of NestedSupertree is that it is not a one off algorithm. Indeed, its main purpose is that it can be used as a platform to construct other supertree methods for rooted semi-labelled trees that are singularly labelled. Moreover, we show in Sections 3 and 4 that any supertree method constructed from this platform satisfies all of the rooted semi-labelled tree analogues of properties (i)—(iii) in the introduction. Furthermore, in Section 5, we describe one way in which NestedSupertree can be used as a platform where the rooted semi-labelled trees in the input are weighted. In addition to (i)—(iii), the resulting algorithm satisfies the rooted semi-labelled tree analogues of (iv) and (v). The restriction to collections of rooted semi-labelled trees that are singularly labelled is for simplicity and functionality (see remarks in Section 3). Indeed, in practice, this is not much of a restriction as rooted semi-labelled trees are typically singularly labelled.

In the last section of the paper, Section 6, we consider what happens when NestedSupertree is applied to a collection $\mathcal{P}$ of rooted phylogenetic trees. In this case, the minor conditions on $\mathcal{P}$ referred to above are redundant and that NestedSupertree applied to $\mathcal{P}$ always returns a rooted phylogenetic tree. Thus NestedSupertree provides a platform for constructing general supertree methods for rooted phylogenetic trees. Furthermore, we show that if $\mathcal{P}$ is compatible, then the rooted phylogenetic tree returned by NestedSupertree is the same as that returned by Build. Thus NestedSupertree is a generalisation of Build. In fact, as we will see, it also generalises AncestralBuild in a corresponding way.

Before ending this section with some preliminaries we make two comments. Firstly, in addition to the properties listed in the introduction, one other property is given in [12]. This property says that "the resulting supertree displays all 'nestings' shared by all of the trees in $\mathcal{P}$", where one subset of the labels in $\mathcal{P}$ *nests in* another if the most recent common ancestor of the former is a strict descendant of the most recent common ancestor of the latter. It has been recently shown by Willson [15] that the proof in [12] that establishes MinCutSupertree has this property is incorrect and, in fact, that MinCutSupertree does not have this property. Whether displaying all shared nestings of the input collection is a desirable property is debatable. We simply note here that NestedSupertree also does not have this property. For the curious reader, there is a general supertree

method for collections $\mathcal{P}$ of rooted phylogenetic trees that satisfies this nesting property as well as the properties listed in the introduction. In particular, first use the BUILD algorithm to either produce a supertree that displays $\mathcal{P}$, in which case the supertree method outputs this tree, or recognise that $\mathcal{P}$ is not compatible. If the latter happens, construct the "Adams consensus tree" $\mathcal{T}$ (see [1] or [13]) for the set $\mathcal{P}'$ of rooted phylogenetic trees obtained from $\mathcal{P}$ by restricting each tree to the subset of labels of $\mathcal{P}$ that are common to each tree in $\mathcal{P}$. This tree displays all of the nestings shared by all of the trees in $\mathcal{P}'$ and hence $\mathcal{P}$. Now, for each remaining label a in $\mathcal{P}$, adjoin a to the root of $\mathcal{T}$ with a distinct new edge. The supertree method outputs the resulting tree. The second comment is that the approach taken by NESTEDSUPERTREE and the approach of MINCUTSUPERTREE are very different. A comparison between these two methods for rooted phylogenetic trees would make an interesting project.

Let $\mathcal{T} = (T; \phi)$ be a rooted semi-labelled tree on X. The set X is called the *label set* of $\mathcal{T}$ and the elements of X are called *labels*. We also use $\mathcal{L}(\mathcal{T})$ to denote the label set of $\mathcal{T}$. If v is a vertex of T, we say that the elements of $\phi^{-1}(v)$ *label* v. Furthermore, $\mathcal{T}$ is *fully labelled* if every vertex of T is labelled by an element of X. For a collection $\mathcal{P}$ of rooted semi-labelled trees, we denote the union of the label sets of the trees in $\mathcal{P}$ by $\mathcal{L}(\mathcal{P})$. Moreover, we call an element x of $\mathcal{L}(\mathcal{P})$ *common* if $x \in \bigcap_{\mathcal{T} \in \mathcal{P}} \mathcal{L}(\mathcal{T})$.

There is a natural and useful partial order on the label set $\mathcal{L}(\mathcal{T})$ of a rooted semi-labelled tree $\mathcal{T} = (T; \phi)$. This partial order is obtained by setting $b \leq_{\mathcal{T}} a$ if the path from the root of T to $\phi^{-1}(a)$ includes $\phi^{-1}(b)$, in which case a we say that a is a *descendant* of b. If $b <_{\mathcal{T}} a$, then we say that a is a *strict descendant* of b. Furthermore, $a, b \in \mathcal{L}(\mathcal{T})$ are *not comparable* under $\leq_{\mathcal{T}}$ if neither $b \leq_{\mathcal{T}} a$ nor $a \leq_{\mathcal{T}} b$ holds.

Lastly, a *rooted triple* is a rooted phylogenetic tree that has two interior vertices and whose label set has size three. We denote the rooted triple $\mathcal{T}$ with label set $\{a, b, c\}$ by $ab|c$ if the path from a to b does not intersect the path from the root to c. For a collection $\mathcal{P}$ of rooted semi-labelled trees, a rooted triple whose label set $\{a, b, c\}$ is a subset of $\bigcap_{\mathcal{T} \in \mathcal{P}} \mathcal{L}(\mathcal{T})$ is *common* relative to $\mathcal{P}$ if, for all $\mathcal{T}_1, \mathcal{T}_2 \in \mathcal{P}$, $\mathcal{T}_1|\{a, b, c\}$ is isomorphic to $\mathcal{T}_2|\{a, b, c\}$. Note that none of a, b, c need label a leaf of $\mathcal{T}_1$ or $\mathcal{T}_2$.

3. THE ALGORITHM NESTEDSUPERTREE

For a collection $\mathcal{P}$ of rooted semi-labelled trees that are singularly labelled, the algorithm NESTEDSUPERTREE applied to $\mathcal{P}$ is based on a particular construction and two graphs. We described the construction first and then the two graphs.

Let $\mathcal{T} = (T; \phi)$ be a rooted semi-labelled tree on X, where T has vertex set V. We say that a rooted fully-labelled tree $\mathcal{T}_1 = (T; \phi_1)$ on X_1, where $X \subseteq X_1$, has been obtained from $\mathcal{T}$ by *adding distinct new labels* if, for all distinct $u, v \in V$, the following properties are satisfied:

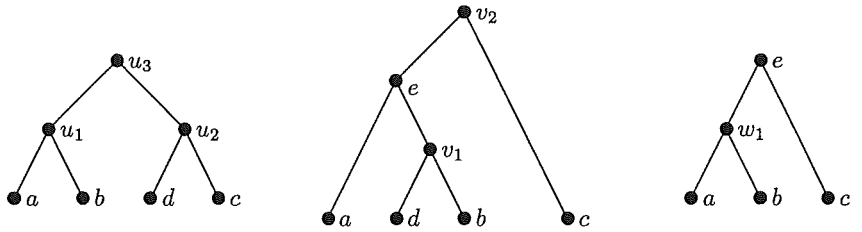


FIGURE 2. A collection $\mathcal{P}'$ of rooted fully-labelled trees.

(1) If $\phi^{-1}(v)$ is non-empty, then $\phi_1^{-1}(v) = \phi^{-1}(v)$.
(2) If $\phi^{-1}(v)$ is empty, then $|\phi_1^{-1}(v)| = 1$.
(3) If $\phi^{-1}(u)$ and $\phi^{-1}(v)$ are both empty, then $\phi_1^{-1}(u) \neq \phi_1^{-1}(v)$.

Intuitively, $\mathcal{T}_1$ has been obtained from $\mathcal{T}$ by adding a distinct new label to each non-labelled vertex of $\mathcal{T}$. For a collection $\mathcal{P}$ of rooted semi-labelled trees, we say that $\mathcal{P}_1$ has been obtained from $\mathcal{P}$ by adding distinct new labels if it has been obtained by adding distinct new labels to each tree in $\mathcal{P}$ so that no pair of added labels are the same.

We now describe the two graphs each of which consists of both arcs (directed edges) and edges. Let $\mathcal{P}$ be a collection of rooted semi-labelled trees and let $\mathcal{P}'$ be a collection of rooted fully-labelled trees obtained from $\mathcal{P}$ by adding distinct new labels. The *descendancy graph of* $\mathcal{P}'$, denoted $D(\mathcal{P}')$, is the graph whose vertex set is $\mathcal{L}(\mathcal{P}')$, whose arc set is

$$\{(c, a) : c <_{\mathcal{T}} a \text{ for some } \mathcal{T} \in \mathcal{P}'\},$$

and whose edge set is

$$\{\{a, b\} : a \text{ is not comparable to } b \text{ under } \leq_{\mathcal{T}} \text{ for some } \mathcal{T} \in \mathcal{P}'\}.$$

The descendancy graph is said to be *acyclic* if, ignoring edges, it has no directed cycles.

The second graph $D'(\mathcal{P}')$ is obtained from the descendancy graph $D(\mathcal{P}')$ of $\mathcal{P}'$ as follows. For each common rooted triple $a_1a_2|b$ of $\mathcal{P}$, add a new vertex $a_1a_2|b$, a new arc from $a_1a_2|b$ to a_1, and a new arc from $a_1a_2|b$ to a_2. Vertices of the form $a_1a_2|b$ are called *rooted triple vertices* of $D'(\mathcal{P}')$; all other vertices of $D'(\mathcal{P}')$ are called *label vertices*. We call $D'(\mathcal{P}')$ the *modified descendancy graph* of $\mathcal{P}'$.

In general, for a graph G that contains both edges and arcs, a vertex of G has *indegree zero* if, ignoring edges, it has no incoming arcs. An *arc component* of G is a component of the graph obtained from G by deleting all edges. Furthermore, for a subset V_1 of the vertex set of G, the restriction of G to V_1 is the subgraph of G that is obtained by deleting all vertices not in V_1 together with their incident edges and arcs. This restriction is denoted by $G|V_1$.

Example 3.1. To illustrate the above construction and graphs, let $\mathcal{P}$ be the collection of rooted semi-labelled trees shown in Fig. 1 and let $\mathcal{P}'$ be the collection of

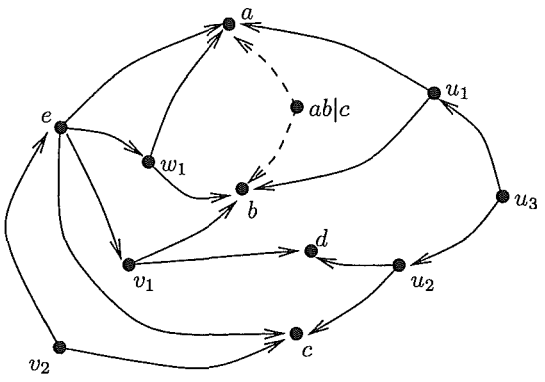


FIGURE 3. The modified descendancy graph of $\mathcal{P}'$.

rooted fully-labelled trees obtained from $\mathcal{P}$ by adding distinct new labels as shown in Fig. 2.

The modified descendancy graph of $\mathcal{P}'$ is shown in Fig. 3 where, for simplicity, the edges as well as the arcs (c, a) where a is not an immediate descendant of c are omitted. If these edges were included, there would, for example, be an edge joining the label vertices w_1 and c as they are not comparable in the rightmost tree of Fig. 2. Furthermore, to highlight the one rooted triple vertex, its outgoing arcs are drawn as dashed arrows. This example will be referred to later in this section and also in Section 5.

Lastly, let $\mathcal{P}$ be a collection of rooted semi-labelled trees that are singularly labelled, and let a and b be elements of $\mathcal{L}(\mathcal{P})$. We say that a and b are *pairwise consistent* if, whenever a is a strict descendant of b in some tree in $\mathcal{P}$, a is always a strict descendant of b in every tree of $\mathcal{P}$ whose label set contains both a and b. Furthermore, $\mathcal{P}$ is said to be *pairwise consistent* if all pairs of labels in $\mathcal{L}(\mathcal{P})$ are pairwise consistent.

We now describe NESTEDSUPERTREE and its subroutine DESCENDANT. An illustrative example and some informative remarks follow these descriptions.

Algorithm: NESTEDSUPERTREE$(\mathcal{P}, \mathcal{T})$
Input: A collection $\mathcal{P}$ of rooted semi-labelled trees that are singularly labelled.
Output: A rooted semi-labelled tree $\mathcal{T}$ with label set $\mathcal{L}(\mathcal{P})$, the statement $\mathcal{P}$ *is not pairwise consistent*, or the statement $\mathcal{P}$ *has an ancestor-descendant contradiction.*

1. For each pair $a, b \in \mathcal{P}$, check that a and b are pairwise consistent. If not, then halt and return $\mathcal{P}$ *is not pairwise consistent.*
2. Construct a collection $\mathcal{P}'$ of rooted fully-labelled trees from $\mathcal{P}$ by adding distinct new labels.
3. Construct the descendancy graph $D(\mathcal{P}')$ of $\mathcal{P}'$.

4. If $D(\mathcal{P}')$ has a directed cycle, then halt and return $\mathcal{P}$ *has an ancestor-descendant contradiction.*
5. Construct the modified descendancy graph $D'(\mathcal{P}')$ of $\mathcal{P}'$.
6. Call the subroutine DESCENDANT$(D'(\mathcal{P}'), v', \mathcal{T}')$.
7. Remove the added labels from $\mathcal{T}'$, suppress any resulting unlabelled degree-two vertex, and relocate the root to the nearest vertex that is either labelled or has outdegree at least two if it is unlabelled and has degree one. Return the resulting rooted semi-labelled tree $\mathcal{T}$.

Algorithm: DESCENDANT$(D'(\mathcal{P}'), v', \mathcal{T}')$
Input: A graph $D'(\mathcal{P}')$.
Output: A rooted fully-labelled tree $\mathcal{T}'$ with root vertex v'.

1. Let $\mathcal{S}_0$ denote the set of label vertices of $D'(\mathcal{P}')$ that have indegree zero and no incident edges.
2. If $\mathcal{S}_0$ is empty, then choose $\mathcal{S}_0$ to be any non-empty subset of label vertices of $D'(\mathcal{P}')$ that have indegree zero.
3. Delete the elements of $\mathcal{S}_0$ (and their incident arcs and their incident edges) from $D'(\mathcal{P}')$. Furthermore, for each common rooted triple $a_1a_2|b$ of $\mathcal{P}$, delete the rooted triple vertex $a_1a_2|b$ if, in the resulting graph, the arc component containing a_1 and a_2 does not contain the label vertex b.
4. Let $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ denote the vertex sets of the arc components of the graph obtained at the end of Step 3.
5. For each element $i \in \{1, 2, \ldots, k\}$, call DESCENDANT$(D'(\mathcal{P}')|\mathcal{S}_i, v_i', \mathcal{T}_i')$. Assign the labels in $\mathcal{S}_0$ to v' and attach $\mathcal{T}_i'$ to v' via the edge $\{v_i', v'\}$.

Example 3.2. As an example of NESTEDSUPERTREE applied to a collection of rooted semi-labelled trees that are singularly labelled, let $\mathcal{P}$ and $\mathcal{P}'$ be the collections described in Example 3.1. On the first iteration of DESCENDANT, the label vertices v_2 and u_3 in the modified descendancy graph $D'(\mathcal{P}')$ have indegree zero and no incident edges, and no other label vertices have this property. Therefore, in this iteration, $\mathcal{S}_0 = \{v_2, u_3\}$. Furthermore, the graph obtain from $D'(\mathcal{P}')$ by deleting the elements of $\mathcal{S}_0$ has exactly one arc component.

In the second iteration, the label vertices of the inputted graph that have indegree zero are e, u_1, and u_2, and each of these have an incident edge. Therefore, in this iteration, we can choose any non-empty subset of $\{e, u_1, u_2\}$ to be $\mathcal{S}_0$. If we choose $\mathcal{S}_0$ to be the whole set, then, in all subsequent iterations of the algorithm, there is always a non-empty set of label vertices of the corresponding graph that have indegree zero and no incident edges. By making this choice, DESCENDANT eventually returns the rooted fully-labelled tree shown in Fig. 4(a) and NESTEDSUPERTREE returns the rooted semi-labelled tree shown in Fig. 4(b).

Remarks.

1. The reason we call NESTEDSUPERTREE a platform is that a choice can be made in Step 2 of DESCENDANT. One possible way of making this choice is described in Section 5. Note that, as we will soon see, if the subroutine is called, but Step 2

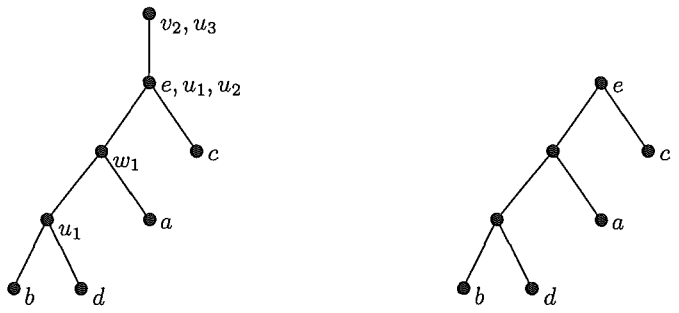


FIGURE 4. (a) One possible output of DESCENDANT when applied to $D'(\mathcal{P}')$ and (b) the corresponding output of NESTEDSUPERTREE.

is never invoked, then the supertree returned by NESTEDSUPERTREE ancestrally displays $\mathcal{P}$.

2. One of the attractions of a general supertree algorithm is that conflicts are resolved in some way so that one always outputs a supertree whether or not the original collection of input trees is compatible. In the case the input is a collection of rooted phylogenetic trees, it is reasonable that any supertree algorithm resolves such conflicts. However, in the case the input is a collection of rooted semi-labelled trees, it appears to us that there are some fundamental ancestor-descendant conflicts that should be resolved separately. Two such conflicts are pairwise consistency and ancestor-descendant contradictions. Finding such conflicts can be easily done in polynomial time. In the case of ancestor-descendant contradictions, see the proof of Lemma 3.4.
3. The purpose of adding the rooted triple vertices and associated arcs to the descendancy graph of $\mathcal{P}'$ is so that any tree outputted by NESTEDSUPERTREE preserves all of the common rooted triples of $\mathcal{P}$. This property and, in particular, the rooted semi-labelled tree analogue of desirable property (ii) is established in the next section.
4. Proposition 3.6 shows that, provided $\mathcal{P}$ is pairwise consistent and the descendancy graph of $\mathcal{P}'$ is acyclic, NESTEDSUPERTREE returns a rooted semi-labelled tree. Thus we can always find a non-empty set $\mathcal{S}_0$ as described in Steps 1 and 2 of the subroutine DESCENDANT.
5. Lastly, the check for the pairwise consistency of $\mathcal{P}$ and the restriction that each tree in the input collection is singularly labelled could be removed from NESTEDSUPERTREE. However, if either is done, then there is no guarantee that the resulting supertree satisfies the rooted semi-labelled tree analogue of (ii) in the introduction.

The rest of this section establishes some basic properties of NESTEDSUPERTREE, in particular, the rooted semi-labelled tree analogues of (i) (Proposition 3.7) and (iii) (Proposition 3.3). Further properties are established in the next section.

We begin by making the following observation. Recall from the introduction that ANCESTRALBUILD is a polynomial-time algorithm that determines if a collection of rooted semi-labelled trees is ancestrally compatible, in which case such a tree is returned [7]. The description of NESTEDSUPERTREE closely resembles the description of ANCESTRALBUILD. Indeed, the latter can be essentially obtained from the former as follows: remove Steps 1, 4 and 5 in NESTEDSUPERTREE; replace the modified descendancy graph of $\mathcal{P}'$ with the descendancy graph of $\mathcal{P}'$ in DESCENDANT; remove the second sentence of Step 3 of DESCENDANT; and, replace Step 2 of DESCENDANT "If $\mathcal{S}_0$ is empty, halt and return $\mathcal{P}'$ *is not ancestrally compatible*", in which case $\mathcal{P}$ is not ancestrally compatible. It follows that NESTEDSUPERTREE can be viewed as a generalisation of ANCESTRALBUILD. Indeed, we have the following proposition.

Proposition 3.3. *Let $\mathcal{P}$ be a collection of rooted semi-labelled trees that are singularly labelled, and suppose that $\mathcal{P}$ is ancestrally compatible. Then* NESTEDSUPERTREE *applied to $\mathcal{P}$ returns a rooted semi-labelled tree that ancestrally displays $\mathcal{P}$.*

Proof. Let $\mathcal{P}'$ be a collection of rooted fully-labelled trees that is obtained from $\mathcal{P}$ by adding distinct new labels. Since $\mathcal{P}$ is ancestrally compatible, $\mathcal{P}$ is pairwise consistent and $D(\mathcal{P}')$ has no directed cycles. It now follows from the description of how ANCESTRALBUILD can be obtained from NESTEDSUPERTREE that NESTEDSUPERTREE applied to $\mathcal{P}$ returns a rooted semi-labelled tree that ancestrally displays $\mathcal{P}$. □

The next two lemmas are needed for the proofs of Propositions 3.6 and 3.7. The first lemma is well-known and an easy exercise. However, we include its proof as it indicates how one can find all of the ancestor-descendant contradictions of a collection of rooted semi-labelled trees.

Lemma 3.4. *Let D be a connected digraph that contains no directed cycle. Then there exists a vertex of D whose indegree is zero.*

Proof. Assume no vertex of D has indegree zero. Let D' be the digraph obtained from D by reversing the orientation of the arcs of D. By assumption, every vertex of D' has outdegree at least one. Let u be a vertex of D'. Starting at u, construct a directed walk. Since each vertex of D' has an outgoing arc, we must eventually meet a vertex on this walk that has already been traversed. In particular, this means that D' contains a directed cycle, which in turn implies that D contains a directed cycle. This contradiction completes the proof of the lemma. □

Lemma 3.5. *Let $\mathcal{P}$ be a collection of rooted fully-labelled trees that are singularly labelled. Let $b \in \bigcap_{\mathcal{T}\in\mathcal{P}} \mathcal{L}(\mathcal{T})$, and let $x, y \in \mathcal{L}(\mathcal{P})$. Suppose that b is pairwise consistent with each of the labels in $\mathcal{L}(\mathcal{P})$. Then the following hold.*

(i) *If there is a directed path from b to x in $D(\mathcal{P})$, then there is an arc from b to x in $D(\mathcal{P})$. Furthermore, if there is a directed path from x to b in $D(\mathcal{P})$, then there is an arc from x to b in $D(\mathcal{P})$.*

(ii) *Suppose that* (b, x) *is an arc in* $D(\mathcal{P})$. *If* (y, x) *is also an arc in* $D(\mathcal{P})$ *and* $b \neq y$, *then either there is an arc from* y *to* b *in* $D(\mathcal{P})$ *or there is an arc from* b *to* y *in* $D(\mathcal{P})$.

Proof. We first prove (i). Assume that $by_1y_2 \cdots y_k x$ is a directed path in $D(\mathcal{P})$ from b to x. As y_1 is a strict descendant of b in some tree in $\mathcal{P}$ and b is pairwise consistent with y_1, it follows that whenever b and y_1 are labels of some tree in $\mathcal{P}$, y_1 is a strict descendant of b. Since there is an arc from y_1 to y_2, there is a tree $\mathcal{T}_1$ in $\mathcal{P}$ in which y_2 is a strict descendant of y_1. Since b is a label of $\mathcal{T}_1$, this implies that y_2 is a strict descendant of b in $\mathcal{T}_1$, so, by definition, there is an arc in $D(\mathcal{P})$ from b to y_2. Repeating this argument for y_2 and y_3, we deduce that there is an arc in $D(\mathcal{P})$ from b to y_3. Continuing in this way, we eventually establish that there is an arc in $D(\mathcal{P})$ from b to x. A similar argument shows that there is an arc (x, b) in $D(\mathcal{P})$ if there is a directed path in $D(\mathcal{P})$ from x to b. This establishes (i).

We now prove (ii). Since (y, x) is an arc of $D(\mathcal{P})$, there is a tree $\mathcal{T}$ in $\mathcal{P}$ for which x is a strict descendant of y. But this means that, as (b, x) is an arc of $D(\mathcal{P})$, b is a common label and is pairwise consistent with x, and all trees in $\mathcal{P}$ are singularly labelled, either b is a strict descendant of y or y is a strict descendant of b in $\mathcal{T}$. In particular, either (b, y) or (y, b) is an arc in $D(\mathcal{P})$, respectively. □

Proposition 3.6. *Let* $\mathcal{P}$ *be a collection of rooted semi-labelled trees that are singularly labelled and let* $\mathcal{P}'$ *be a collection of fully-labelled trees obtained from* $\mathcal{P}$ *by adding distinct new labels. If* $\mathcal{P}$ *is pairwise consistent and the descendancy graph of* $\mathcal{P}'$ *is acyclic, then* NESTEDSUPERTREE *applied to* $\mathcal{P}$ *returns a rooted semi-labelled tree.*

Proof. Because of Lemma 3.4 and the fact that DESCENDANT successively considers proper restrictions of the modified descendancy graph of $\mathcal{P}'$, it suffices to show that one can always choose a non-empty set $\mathcal{S}_0$ of label vertices in Steps 1 and 2 at each iteration of the subroutine DESCENDANT. To see this, suppose that at some iteration of DESCENDANT the associated connected restriction, D say, of $D'(\mathcal{P}')$ has no label vertex of indegree zero. Let $\mathcal{S}$ be the set of label vertices of D in which the only incoming arcs are the ones coming from rooted triple vertices. Since any restriction of the descendancy graph of $\mathcal{P}'$ is acyclic, it follows that $\mathcal{S}$ is non-empty. Let a_1 be an element of $\mathcal{S}$. By the construction of the modified descendancy graph, a_1 is a label of a common rooted triple, $a_1a_2|b$ say, of $\mathcal{P}$. Furthermore, a_1 is not the only element of $\mathcal{S}$; for otherwise, every label vertex of D, including a_2, would be a strict descendant of a_1. It now follows that, as D is connected, there is a label vertex w that lies in a directed path from a_1 and that also lies in a directed path from a common label vertex, x_1 say, that is distinct from a_1 and is in $\mathcal{S}$. By Lemma 3.5(i), this implies that there exists a tree $\mathcal{T}_1$ in $\mathcal{P}$ in which w is a strict descendant of a_1 and a tree $\mathcal{T}_2$ in $\mathcal{P}$ in which w is a strict descendant of x_1. As a_1 and x_1 are not comparable in $\mathcal{T}_1$, it follows that w is not comparable to x_1 in $\mathcal{T}_1$. But w is a strict descendant of x_1 in $\mathcal{T}_2$, contradicting the assumption that $\mathcal{P}$ is pairwise consistent. We conclude that at Steps 1 and 2 of each iteration of DESCENDANT, we can always find an appropriate non-empty set of label vertices. □

Proposition 3.7. *Let $\mathcal{P}$ be a collection of rooted semi-labelled trees that are singularly labelled. Then the running time of* NESTEDSUPERTREE *applied to $\mathcal{P}$ is polynomial in $|\mathcal{L}(\mathcal{P})| \times |\mathcal{P}|$.*

Proof. Let $\mathcal{P}'$ be a collection of rooted fully-labelled trees that is obtained from $\mathcal{P}$ by adding distinct new labels. Since the only possible unlabelled vertices of a rooted semi-labelled tree are either the root vertex or a vertex of degree at least three, the number of such interior vertices is at most one less than the number of leaves. Therefore, to prove the proposition, it suffices to show that the running time of NESTEDSUPERTREE is polynomial in $|\mathcal{L}(\mathcal{P}')| \times |\mathcal{P}|$.

It is clear that checking for pairwise consistency is polynomial time in $|\mathcal{L}(\mathcal{P}')| \times |\mathcal{P}|$. Furthermore, the construction of the descendancy graph of $\mathcal{P}'$ can be also be done in such a time. Now one can determine if a directed graph has no directed cycles by successively deleting vertices (and their incident arcs) that either have indegree or outdegree zero. If this process results in the empty graph, then the original graph has no directed cycles; otherwise it has a directed cycle. Since the size of $D(\mathcal{P}')$ is polynomial in the size of $\mathcal{L}(\mathcal{P}')$, determining whether or not $D(\mathcal{P}')$ has no directed cycles is polynomial in the size of $\mathcal{L}(\mathcal{P}')$.

The number of triples of $\mathcal{L}(\mathcal{P})$ is polynomial in $|\mathcal{L}(\mathcal{P})|$ and so finding the collection of common rooted triples of $\mathcal{P}$ is also polynomial in $|\mathcal{L}(\mathcal{P}')| \times |\mathcal{P}|$. It follows that the construction of the modified descendancy graph of $\mathcal{P}'$ is polynomial time in $|\mathcal{L}(\mathcal{P}')| \times |\mathcal{P}|$. Lastly, at each iteration of the subroutine DESCENDANT, we successively consider proper restrictions of $D'(\mathcal{P}')$ and so the number of such iterations is bounded by the size of $\mathcal{L}(\mathcal{P}')$. We deduce that the running time of NESTEDSUPERTREE is polynomial in $|\mathcal{L}(\mathcal{P}')| \times |\mathcal{P}|$. □

4. OTHER PROPERTIES OF NESTEDSUPERTREE

The main purpose of this section is to establish the rooted semi-labelled tree analogue of desirable property (ii) in the introduction for NESTEDSUPERTREE.

A rooted semi-labelled tree $\mathcal{T}$ is *binary* if $\mathcal{T}$ is singularly labelled and every vertex has degree at most three except for the root which has degree at most two. The main result of this section is the following theorem.

Theorem 4.1. *Let $\mathcal{P}$ be a collection of semi-labelled trees that are singularly labelled and let $\mathcal{T}$ be a rooted semi-labelled binary tree that is ancestrally displayed by each tree in $\mathcal{P}$. Suppose that* NESTEDSUPERTREE *applied to $\mathcal{P}$ returns a rooted semi-labelled tree $\mathcal{T}'$. Then $\mathcal{T}'$ ancestrally displays $\mathcal{T}$.*

To prove Theorem 4.1, we first establish several results. The first result, Proposition 4.2, is well-known (for example, see [13]).

Proposition 4.2. *Let $\mathcal{T}$ be a rooted phylogenetic X-tree. Let*

$$\mathcal{R}(\mathcal{T}) = \{\mathcal{T}|S : S \subseteq X,\ |S| = 3,\ \mathcal{T}|S \text{ is a rooted triple}\}.$$

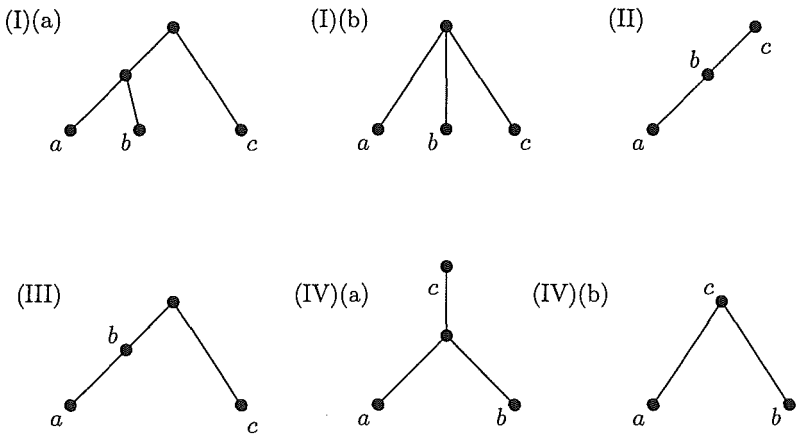


FIGURE 5. The six triples.

If T' is a rooted phylogenetic X'-tree, where $X \subseteq X'$, and $\mathcal{R}(T) \subseteq \mathcal{R}(T')$, then T' displays T.

For rooted semi-labelled trees that are singularly labelled, the analogous result is Proposition 4.3. A *triple* is a rooted semi-labelled tree that is singularly labelled and has label set of size three. A rooted triple is a particular type of triple. Up to isomorphism, there are six triples and these are shown in Fig. 5. For convenience in this paper, we denote these triples as Types (I)(a) and (b), (II), (III), and (IV)(a) and (b). We will continue to refer to a triple of Type(I)(a) as a rooted triple.

Proposition 4.3. *Let T be a rooted semi-labelled tree on X. Let*

$$\mathcal{B}(T) = \{T|S : S \subseteq X,\ |S| = 3,\ T|S \text{ is a triple of Type (I)(a) or (IV)(a)}\},$$

$$\mathcal{D}(T) = \{c <_T a : a, c \in \mathcal{L}(T)\},$$

and

$$\mathcal{N}(T) = \{a \text{ is not comparable to } b \text{ under } \leq_T : a, b \in \mathcal{L}(T)\}.$$

If T' is a rooted semi-labelled tree on X', where $X \subseteq X'$, and $\mathcal{B}(T) \subseteq \mathcal{B}(T')$, $\mathcal{D}(T) \subseteq \mathcal{D}(T')$, and $\mathcal{N}(T) \subseteq \mathcal{N}(T')$, then T' ancestrally displays T.

Proof. To prove the proposition, it is clear that we may assume that X and X' are the same sets. Let $\mathcal{T} = (T; \phi)$. The proof is by induction on the number n of interior labels of $\mathcal{T}$. If $n = 0$, then it is straightforward to deduce the result by Proposition 4.2 and the fact that $\mathcal{N}(T) \subseteq \mathcal{N}(T')$. Now assume that the result holds for all rooted semi-labelled trees that have fewer than n interior labels, where $n \geq 1$. Since $\mathcal{T}$ has at least one interior label, there exists an interior vertex u of T that is labelled by an element, d say, of X such that all elements of X that are strict descendants of d label leaves of $\mathcal{T}$. Let $\mathcal{T}_1$ be the rooted semi-labelled tree obtained from $\mathcal{T}$ by replacing the rooted subtree of $\mathcal{T}$ that lies below or equal to u with a single leaf labelled by the elements of $\phi^{-1}(u)$. Let $\mathcal{T}_2$ be the rooted semi-labelled tree that is the rooted subtree of $\mathcal{T}$ that lies below or equal to d and

in which the elements in $\phi^{-1}(u)$ are removed. Note that if u has 'outdegree' one, then u is deleted and the root of $\mathcal{T}_2$ is the vertex of $\mathcal{T}$ that is immediately below u.

Now consider $\mathcal{T}'$. Since $\mathcal{D}(\mathcal{T}) \subseteq \mathcal{D}(\mathcal{T}')$, each element in $\phi^{-1}(u)$ labels an interior vertex of $\mathcal{T}'$. Moreover, as there is an element of X that is a strict descendant of each element in $\phi^{-1}(u)$, it follows that, for all pairs $a, b \in \phi^{-1}(u)$, either a and b label the same vertex of $\mathcal{T}'$ or one element, a say, is a strict descendant of b in $\mathcal{T}'$. Let c be a least element of $\phi^{-1}(u)$ under $\leq_{\mathcal{T}'}$ and let v be the interior vertex of $\mathcal{T}'$ that is labelled by c. Again, as $\mathcal{D}(\mathcal{T}) \subseteq \mathcal{D}(\mathcal{T}')$, the set of strict descendants of c in $\mathcal{T}'$ is exactly the label set of $\mathcal{T}_2$. Analogous to the constructions of $\mathcal{T}_1$ and $\mathcal{T}_2$ in the previous paragraph, construct $\mathcal{T}_1'$ and $\mathcal{T}_2'$ from $\mathcal{T}'$ using the vertex v instead of u. Evidently, $\mathcal{B}(\mathcal{T}_1) \subseteq \mathcal{B}(\mathcal{T}_1')$, $\mathcal{D}(\mathcal{T}_1) \subseteq \mathcal{D}(\mathcal{T}_1')$, and $\mathcal{N}(\mathcal{T}_1) \subseteq \mathcal{N}(\mathcal{T}_1')$, and $\mathcal{B}(\mathcal{T}_2) \subseteq \mathcal{B}(\mathcal{T}_2')$, $\mathcal{D}(\mathcal{T}_2) \subseteq \mathcal{D}(\mathcal{T}_2')$, and $\mathcal{N}(\mathcal{T}_2) \subseteq \mathcal{N}(\mathcal{T}_2')$. Furthermore, both $\mathcal{T}_1$ and $\mathcal{T}_2$ have fewer than n labelled interior vertices. Therefore, by our induction assumption, $\mathcal{T}_1'$ and $\mathcal{T}_2'$ ancestrally display $\mathcal{T}_1$ and $\mathcal{T}_2$, respectively. By definition, it immediately follows that $\mathcal{T}'$ ancestrally displays $\mathcal{T}$ unless u has 'outdegree' one and the vertex of $\mathcal{T}'$ labelled by c has 'outdegree' at least two. But then, in this case, there are elements $a, b \in X$ such that $\mathcal{T}|\{a, b, c\}$ is of Type (IV)(a) and $\mathcal{T}'|\{a, b, c\}$ is of Type (IV)(b), contradicting the assumptions in the statement of the proposition. This completes the proof of Proposition 4.3. □

Lemma 4.4. *Let $\mathcal{P}$ be a collection of rooted semi-labelled trees that are singularly labelled and let $a, b \in \mathcal{L}(\mathcal{P})$. Suppose that* NESTEDSUPERTREE *applied to $\mathcal{P}$ returns a rooted semi-labelled tree $\mathcal{T}$.*

(i) *If a is a strict descendant of b in some tree in $\mathcal{P}$, then a is a strict descendant of b in $\mathcal{T}$.*
(ii) *If $a, b \in \bigcap_{T \in \mathcal{P}} \mathcal{L}(T)$, and a is not comparable to b in each tree in $\mathcal{P}$, then a is not comparable to b in $\mathcal{T}$.*

Proof. Part (i) immediately follows from the description of NESTEDSUPERTREE.

To prove (ii), let $\mathcal{P}'$ be the collection of rooted fully-labelled trees that is obtained from $\mathcal{P}$ by adding distinct new labels in Step 2 of NESTEDSUPERTREE. At some iteration of the running of the sub-routine DESCENDANT, one of the label vertices a or b in some restriction of the modified descendancy graph $D'(\mathcal{P}')$ of $\mathcal{P}'$ has indegree zero. Consider the first such iteration and let D denote the corresponding connected graph. Without loss of generality, we may assume that a has indegree zero in this restriction. To establish the lemma, it suffices to show by the construction of $\mathcal{T}$ that b is not a vertex of D.

Let V_a be the subset of vertices of D that are either label vertices lying on a directed path starting at a or rooted triple vertices where both adjacent label vertices lie on a directed path starting at a. Since a and b are not comparable in every tree in $\mathcal{P}$ and a has indegree zero, it follows by the contrapositive of Lemma 3.5(i) that $b \notin V_a$. Thus to establish that b is not a vertex of D, it suffices to show that V_a is the vertex set of D. To see this, suppose that D contains an arc (z, x) where $z \notin V_a$, but $x \in V_a$. Clearly, x is a label vertex of D. Assume that z is also a label vertex of D. By Lemma 3.5(i), (a, x) is an arc of D. Therefore, as a

has indegree zero, it follows by Lemma 3.5(ii) that there is an arc from a to z in D. This implies that $z \in V_a$; a contradiction. In fact, by extending this argument, it is easily seen that, ignoring rooted triple vertices, V_a contains all of the label vertices of D. It now follows by the definition of V_a that V_a is the vertex set of D. This completes the proof of the lemma. □

We remark here that the condition a and b are common labels of $\mathcal{P}$ in the statement of Lemma 4.4(ii) cannot be weakened.

Let $\mathcal{P}$ be a collection of rooted semi-labelled trees. A triple whose label set $\{a, b, c\}$ is a subset of $\bigcap_{\mathcal{T} \in \mathcal{P}} \mathcal{L}(\mathcal{T})$ is *common* relative to $\mathcal{P}$ if, for all $\mathcal{T}_1, \mathcal{T}_2 \in \mathcal{P}$, $\mathcal{T}_1|\{a, b, c\}$ is isomorphic to $\mathcal{T}_2|\{a, b, c\}$.

Lemma 4.5. *Let $\mathcal{P}$ be a collection of rooted semi-labelled trees that are singularly labelled, and let $\mathcal{T}$ be a common triple of $\mathcal{P}$ of Type (I)(a) or (IV)(a). Let $\{a, b, c\}$ be the label set of $\mathcal{T}$. Suppose that* NESTEDSUPERTREE *applied to $\mathcal{P}$ returns a rooted semi-labelled tree $\mathcal{T}'$. Then $\mathcal{T}'|\{a, b, c\}$ is isomorphic to $\mathcal{T}$.*

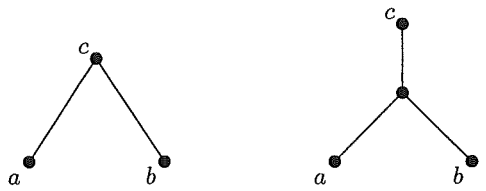
Proof. If $\mathcal{T}$ is a triple of Type (IV)(a), then it is easily seen, by interpreting Lemma 4.4 for a collection of rooted fully-labelled trees that are singularly labelled, that $\mathcal{T}'|\{a, b, c\}$ is isomorphic to $\mathcal{T}$. Therefore suppose that $\mathcal{T}$ is the rooted triple $ab|c$ say. Let $\mathcal{P}'$ be the collection of rooted fully-labelled trees that is obtained from $\mathcal{P}$ by adding distinct new labels in Step 2 of NESTEDSUPERTREE. Since a, b, and c are common labels of $\mathcal{P}$, it follows by Lemma 4.4 that every pair of a, b, and c are not comparable in $\mathcal{T}$. Furthermore, by Step 3 of DESCENDANT, a and b are always in the same arc component of the restrictions of the modified descendancy graph of $\mathcal{P}'$ that are considered throughout the running of NESTEDSUPERTREE provided c is in the same restriction. We now deduce from the description of DESCENDANT that $\mathcal{T}'|\{a, b, c\}$ is isomorphic to $ab|c$. □

We now prove Theorem 4.1.

Proof of Theorem 4.1. Since each label of $\mathcal{T}$ is a common label of $\mathcal{P}$, it immediately follows by Lemma 4.4 that $\mathcal{D}(\mathcal{T}) \subseteq \mathcal{D}(\mathcal{T}')$ and $\mathcal{N}(\mathcal{T}) \subseteq N(\mathcal{T}')$. Furthermore, by Lemma 4.5, $\mathcal{B}(\mathcal{T}) \subseteq \mathcal{B}(\mathcal{T}')$. Hence, by Proposition 4.3, $\mathcal{T}'$ ancestrally displays $\mathcal{T}$. □

Corollary 4.6. *Let $\mathcal{P}$ be a collection of rooted semi-labelled trees that are singularly labelled. Suppose that* NESTEDSUPERTREE *applied to $\mathcal{P}$ returns a rooted semi-labelled tree $\mathcal{T}'$. Then*

(i) *If $\mathcal{T}$ is a common triple of $\mathcal{P}$, then $\mathcal{T}'$ ancestrally displays $\mathcal{T}$.*
(ii) *Let $\{a, b, c\}$ be a subset of $\bigcap_{\mathcal{T} \in \mathcal{P}} \mathcal{L}(\mathcal{T})$. Suppose that, for all $\mathcal{T} \in \mathcal{P}$, $\mathcal{T}|\{a, b, c\}$ is one of the two triples shown in Fig. 6. Then $\mathcal{T}'$ ancestrally displays the triple shown in Fig. 6(b).*



(a) (b)

FIGURE 6. Two triples.

Proof. If $\mathcal{T}$ is a common triple of any type except Type (I)(b), then (i) follows from Theorem 4.1. If $\mathcal{T}$ is a common triple of Type (I)(b), then (i) follows from Lemma 4.4(ii).

For (ii), a routine check using both parts of Lemma 4.4 establishes this part of the corollary. □

We end this section with an observation regarding the last corollary. Observe that for the two triples in (ii) of this corollary one is a refinement of the other. Amongst the other triples only one other pair as this property, Types (I)(a) and (b). Despite part (ii) of Corollary 4.6, it is straightforward to construct an example where the analogue of (ii) for Types (I)(a) and (I)(b) does not hold. This is not a weakness of NESTEDSUPERTREE, but simply highlights the fact shown in [14] that no general supertree method for rooted phylogenetic trees (and hence rooted semi-labelled trees) is able to satisfy this analogue.

5. A SUPERTREE METHOD BASED ON NESTEDSUPERTREE

In this section, we present a supertree algorithm for rooted semi-labelled trees based on the NESTEDSUPERTREE platform. This algorithm, which we call ANCESTRALSUPERTREE, allows the inputed trees to be weighted and also satisfies the symmetry properties of ordering and renaming.

Algorithm: ANCESTRALSUPERTREE($\mathcal{P}, \mathcal{T}$)
Input: A collection $\mathcal{P}$ of rooted semi-labelled trees that are singularly labelled and weighted.
Output: A rooted semi-labelled tree $\mathcal{T}$ with label set $\mathcal{L}(\mathcal{P})$, the statement $\mathcal{P}$ *is not pairwise consistent*, or the statement $\mathcal{P}$ *has an ancestor-descendant contradiction.*

1. For each pair $a, b \in \mathcal{P}$, check that a and b are pairwise consistent. If not, then halt and return $\mathcal{P}$ *is not pairwise consistent.*
2. Construct a collection $\mathcal{P}'$ of rooted fully-labelled trees from $\mathcal{P}$ by adding distinct new labels. Weight the trees in $\mathcal{P}'$ with the corresponding weightings of the trees in $\mathcal{P}$.
3. Construct the descendancy graph $D(\mathcal{P}')$ of $\mathcal{P}'$.

4. If $D(\mathcal{P}')$ has a directed cycle, then halt and return $\mathcal{P}$ *has an ancestor-descendant contradiction.*
5. Construct the modified descendancy graph $D'(\mathcal{P}')$ of $\mathcal{P}'$.
6. Call the subroutine DESCENDANTSUPERTREE$(D'(\mathcal{P}'), v', \mathcal{T}')$.
7. Remove the added labels from $\mathcal{T}'$, suppress any resulting unlabelled degree-two vertex, and relocate the root to the nearest vertex that is either labelled or has outdegree at least two if it is unlabelled and has degree one. Return the resulting rooted semi-labelled tree $\mathcal{T}$.

Algorithm: DESCENDANTSUPERTREE$(D'(\mathcal{P}'), \mathcal{P}', v', \mathcal{T}')$
Input: A graph $D'(\mathcal{P}')$ and a collection $\mathcal{P}'$ of rooted fully-labelled trees in which each tree is weighted.
Output: A rooted fully-labelled tree $\mathcal{T}'$ with root vertex v'.

1. Let $\mathcal{S}_0$ denote the set of label vertices of $D'(\mathcal{P}')$ that have indegree zero and no incident edges.
2. If $\mathcal{S}_0$ is empty, then choose $\mathcal{S}_0$ as follows:
   (a) Let $\mathcal{C}_0$ denote the set of label vertices of $D'(\mathcal{P}')$ that have indegree zero.
   (b) For each $c \in \mathcal{C}_0$, weight c to be the sum of the weights of the trees in $\mathcal{P}'$ that induce at least one incident edge with c in $D'(\mathcal{P}')$.
   (c) Let $\mathcal{S}_0$ consist of the elements of $\mathcal{C}_0$ with minimum weight.
3. Delete the elements of $\mathcal{S}_0$ (and their incident arcs and their incident edges) from $D'(\mathcal{P}')$. Furthermore, for each common rooted triple $a_1a_2|b$ of $\mathcal{P}$, delete the rooted triple vertex $a_1a_2|b$ if, in the resulting graph, the arc component containing a_1 and a_2 does not contain the label vertex b.
4. Let $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ denote the vertex sets of the arc components of the graph obtained at the end of Step 3.
5. For each $i \in \{1, 2, \ldots, k\}$, call DESCENDANTSUPERTREE$(D'(\mathcal{P}')|\mathcal{S}_i, \mathcal{P}', v_i', \mathcal{T}_i')$. Assign the labels in $\mathcal{S}_0$ to v' and attach $\mathcal{T}_i'$ to v' via the edge $\{v_i', v'\}$.

Remarks

1. Clearly, at each iteration of DESCENDANTSUPERTREE, $\mathcal{S}_0$ is non-empty either at the end of Step 1 or Step 2. Furthermore, the time taken to find $\mathcal{S}_0$ is polynomial in $|\mathcal{L}(\mathcal{P})| \times |\mathcal{P}|$. It immediately follows by the results established in Sections 3 and 4 for NESTEDSUPERTREE that ANCESTRALSUPERTREE applied to a collection of rooted semi-labelled trees that are singularly labelled and weighted satisfies the rooted semi-labelled tree analogues of (i)—(iii) and (v) in the introduction.
2. In comparison with DESCENDANT, the set $\mathcal{S}_0$ of label vertices is well-defined in the subroutine ANCESTRALSUPERTREE. Since no appeal is made to the specific symbols used as labels or to the order in which the members of $\mathcal{P}$ are listed in ANCESTRALSUPERTREE, it follows that ANCESTRALSUPERTREE also satisfies the rooted semi-labelled tree analogues of (iv)(a) and (b) in the introduction.

Example 5.1. To illustrate ANCESTRALSUPERTREE, consider the collection $\mathcal{P}$ of rooted semi-labelled trees described in Example 3.1 and the collection $\mathcal{P}'$ of rooted fully-labelled trees obtained from $\mathcal{P}$ by adding distinct new labels. For the purposes of the example, suppose that the three trees in Fig. 1 are weighted so that the

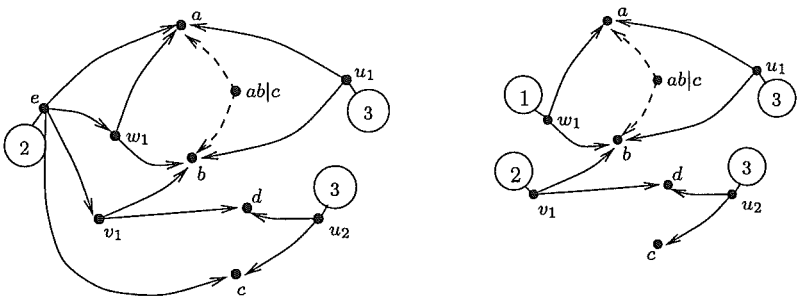


FIGURE 7. The associated graphs in the second and third iteration of DESCENDANTSUPERTREE in Example 5.1.

leftmost tree is weighted 3, the middle tree is weighted 2, and the rightmost tree is weighted 1. The modified descendancy graph of $\mathcal{P}'$ is the same as that given in Fig. 3.

Applying ANCESTRALSUPERTREE to $\mathcal{P}$, the first iteration of DESCENDANTSUPERTREE is the same as that in Example 3.2. In particular, $\mathcal{S}_0 = \{v_2, u_3\}$ at the end of Step 1 and so, in this iteration, no label vertices of the inputted graph are weighted. In the second iteration of DESCENDANTSUPERTREE, $\mathcal{S}_0$ is empty after Step 1. At Step 2(a), the set $\mathcal{C}_0$ of label vertices of the inputted graph with no incoming arcs is $\{e, u_1, u_2\}$. Since the label vertex e has exactly one incident edge and this is induced by the tree with weight 2, we give e weight 2 at Step 2(b) in this iteration. Similarly, u_1 and u_2 are both weighted 3. This weighting together with the associated graph is shown in Fig. 7(a), where the edges and the arcs (c, a) in which a is not an immediate descendant of c are omitted. At Step 2(c), $\mathcal{S}_0 = \{e\}$ and so, at this iteration, it is e and its incident arcs and edges that are deleted from the input graph. The graph resulting from these deletions is shown in Fig. 7(b), where the weights of the label vertices with indegree zero are also shown. Continuing in this way, DESCENDANTSUPERTREE eventually returns the rooted fully-labelled tree shown Fig. 8(a) and ANCESTRALSUPERTREE returns the rooted semi-labelled tree shown in Figure 8(b).

Remarks. Although we think ANCESTRALSUPERTREE is a reasonable algorithm, we expect there to be more elaborate algorithms for supertree construction based on NESTEDSUPERTREE. The point is that it highlights how one can use NESTEDSUPERTREE as a platform for constructing new supertree methods for rooted semi-labelled trees that satisfy all of the rooted semi-labelled tree analogues of the properties listed in the introduction.

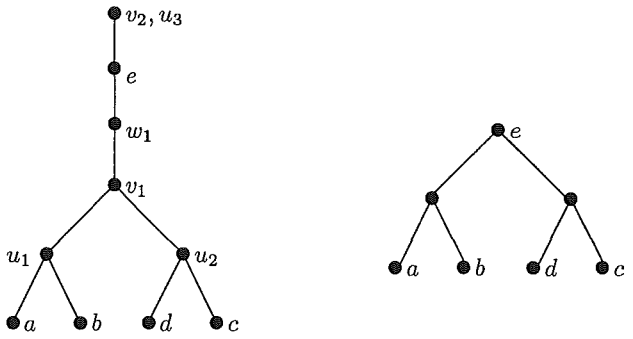


FIGURE 8. The trees returned by DESCENDANTSUPERTREE and ANCESTRALSUPERTREE in Example 5.1.

6. NESTEDSUPERTREE APPLIED TO ROOTED PHYLOGENETIC TREES

Although not originally intended for phylogenetics, the algorithm BUILD [2] was one of the first supertree methods for collections $\mathcal{P}$ of rooted phylogenetic trees. Furthermore, as well as MINCUTSUPERTREE and its modified version, the general approach taken by BUILD has been used in a number of more recent supertree algorithms, for example [5, 6, 8, 11]. In the setting of phylogenetics, BUILD is a polynomial-time algorithm for deciding if $\mathcal{P}$ is compatible. In this section, we describe how NESTEDSUPERTREE can be applied to $\mathcal{P}$ to determine the compatibility of $\mathcal{P}$. In the case that $\mathcal{P}$ is compatible, we also show that the rooted phylogenetic tree returned by NESTEDSUPERTREE is the same as that returned by BUILD.

Since a collection $\mathcal{P}$ of rooted phylogenetic trees is compatible if and only if it is ancestrally compatible, it follows by the discussion prior to Proposition 3.3 that NESTEDSUPERTREE can be suitably modified to determine the compatibility of $\mathcal{P}$. Theorem 6.1 shows that, when applied to the same collection of compatible rooted phylogenetic trees, the supertrees returned by NESTEDSUPERTREE with this modification and BUILD are identical up to isomorphism.

Before stating Theorem 6.1, we first give a description of BUILD. Let $\mathcal{P}$ be a collection of rooted phylogenetic trees and let $\mathcal{S}$ be a subset of $\mathcal{L}(\mathcal{P})$. Let $[\mathcal{P}, \mathcal{S}]$ be the graph that has vertex set S and has an edge joining two vertices a and b precisely if there exists a $c \in \mathcal{S}$ and a $\mathcal{T} \in \mathcal{P}$ such that

$$\mathcal{T}|\{a, b, c\} \cong ab|c.$$

Algorithm: BUILD$(\mathcal{P}, v, \mathcal{T})$
Input: A collection $\mathcal{P}$ of rooted phylogenetic trees.
Output: A rooted phylogenetic tree $\mathcal{T}$ that displays $\mathcal{P}$ with root vertex v, or the statement $\mathcal{P}$ *is not compatible.*

1. Set $\mathcal{S}$ to be the label set of $\mathcal{P}$.
2. If $|\mathcal{S}| = 1$, then output the rooted phylogenetic tree consisting of the single vertex v labelled by the element in $\mathcal{S}$.
3. If $|\mathcal{S}| \geq 2$, construct $[\mathcal{P}, \mathcal{S}]$.
4. Let $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_k$ denote the vertex sets of the components of $[\mathcal{P}, \mathcal{S}]$. If $k = 1$, then halt and return $\mathcal{P}$ *is not compatible.*
5. For each $i \in \{1, 2, \ldots, k\}$, call BUILD$(\mathcal{P}_i, v_i, \mathcal{T}_i)$, where $\mathcal{P}_i$ is the collection of rooted phylogenetic trees obtained from $\mathcal{P}$ by restricting each tree in $\mathcal{P}$ to $\mathcal{S}_i$. If BUILD$(\mathcal{P}_i, v_i, \mathcal{T}_i)$ returns a tree, then attach $\mathcal{T}_i$ to v via the edge $\{v_i, v\}$.

Theorem 6.1. *Let $\mathcal{P}$ be a collection of rooted phylogenetic trees, and suppose that $\mathcal{P}$ is compatible. Then, up to isomorphism, the rooted phylogenetic trees returned by* NESTEDSUPERTREE *with the above modifications and* BUILD *when applied to $\mathcal{P}$ are identical.*

Proof. We begin the proof with two observations. Let $\mathcal{S}$ denote the label set of $\mathcal{P}$, and let $\mathcal{P}'$ be a collection of rooted fully-labelled trees that is obtained from $\mathcal{P}$ by adding distinct new labels. The first observation is that the vertex set of each component of the graph $[\mathcal{P}, \mathcal{S}]$ is a union of maximal proper clusters of the trees in $\mathcal{P}$. For the second observation, consider the descendancy graph of $\mathcal{P}'$, and let $\mathcal{S}_0$ denote the set of vertices of $D(\mathcal{P}')$ that have indegree zero and no incident edges. Then the vertex sets of each arc component of $D(\mathcal{P}')\backslash\mathcal{S}_0$ is also a union of maximal proper clusters of the trees in $\mathcal{P}'$. From these two observations, it is easily deduced, for all $a, b \in \mathcal{S}$, that a and b are in the same component of $[\mathcal{P}, \mathcal{S}]$ if and only if a and b are in the same arc component of $D(\mathcal{P}')\backslash\mathcal{S}_0$.

Let $\mathcal{S}_i$ be the vertex set of a component of $[\mathcal{P}, \mathcal{S}]$ and let $\mathcal{S}_i'$ be the vertex set of the arc component of $D(\mathcal{P}')\backslash\mathcal{S}_0$ that contains $\mathcal{S}_i$. Let $\mathcal{P}_i$ be the collection of rooted phylogenetic trees obtained from $\mathcal{P}$ by restricting each tree in $\mathcal{P}$ to $\mathcal{S}_i$, and let $\mathcal{P}_i'$ be the collection of rooted semi-labelled trees obtained from $\mathcal{P}'$ by restricting each tree in $\mathcal{P}'$ to $\mathcal{S}_i'$. It is easily seen that all of the trees in $\mathcal{P}_i'$ are fully labelled. Now the equivalence at the end of the last paragraph implies that $\mathcal{P}_i'$ could have been obtained from $\mathcal{P}_i$ by adding distinct new labels. Furthermore, the arc component of $D(\mathcal{P}')$ containing the elements of $\mathcal{S}_i$ is equal to the descendancy graph of $D(\mathcal{P}_i')$. Since $[\mathcal{P}, \mathcal{S}]$ contains at least two components, this implies that the maximal proper clusters of the trees returned by NESTEDSUPERTREE with the appropriate modifications and BUILD when applied to $\mathcal{P}$ are the same. Repeatedly applying this argument to $\mathcal{P}_i$ for all i, we eventually deduce that the two rooted phylogenetic trees returned by NESTEDSUPERTREE with the appropriate modifications and BUILD are identical. □

We end this section by remarking on what happens when NESTEDSUPERTREE is applied to an arbitrary collection $\mathcal{P}$ of rooted phylogenetic trees. Let $\mathcal{P}'$ be a collection of rooted fully-labelled trees that is obtained from $\mathcal{P}$ by adding distinct new labels. Since each of the trees in $\mathcal{P}$ are phylogenetic, $\mathcal{P}$ is pairwise consistent and the descendancy graph of $\mathcal{P}'$ is acyclic. It is now easily seen from the description of DESCENDANT that NESTEDSUPERTREE applied to $\mathcal{P}$ returns a rooted semi-labelled tree and that this tree is phylogenetic. It now follows by Propositions 3.7

and 3.3, and Theorem 4.1 that NESTEDSUPERTREE is a general supertree method for rooted phylogenetic trees that satisfies the desirable properties (i)—(iii) in the introduction.

REFERENCES

[1] E. N. Adams III (1986), N-trees as nestings: complexity, similarity and consensus, J. Classification, 3, pp. 299-317.

[2] A. V. Aho, Y. Sagiv, T. G. Szymanski, and J. D. Ullman (1981), Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions, SIAM J. Comput., 10, pp. 405-421.

[3] O. R. P. Bininda-Emonds, J. L. Gittleman, and M. A. Steel (2002), The (super)tree of life: procedures, problems and prospects, Annual Reviews of Ecology and Systematics, 33, pp. 265-289.

[4] O. R. P. Bininda-Emonds, ed., Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life, Computational Biology Series, Kluwer, in press.

[5] D. Bryant, C. Semple, and M. Steel, Combining evolutionary trees with ancestral divergence dates, in Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life, O. R. P. Bininda-Emonds, ed., Computational Biology Series, Kluwer, in press.

[6] M. Constantinescu and D. Sankoff (1995), An efficient algorithm for supertrees, J. Classification, 12, pp. 101-112.

[7] P. Daniel and C. Semple, Supertree algorithms for nested taxa, in Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life, O. R. P. Bininda-Emonds, ed., Computational Biology Series, Kluwer, in press.

[8] M. P. Ng and N. C. Wormald (1996), Reconstruction of rooted trees from subtrees, Discrete Appl. Math., 69, pp. 19-31.

[9] R. D. M. Page, Modified mincut supertrees, in Proceedings of the Second International Workshop on Algorithms in Bioinformatics (WABI 2002), R. Guig and D. Gusfield, eds, Springer, 2002, pp. 537-552.

[10] R. D. M. Page, Taxonomy, supertrees, and the Tree of Life, in Phylogenetic Supertrees: Combining Information to Reveal the Tree of Life, O. R. P. Bininda-Emonds, ed., Computational Biology Series, Kluwer, in press.

[11] C. Semple (2003), Reconstructing minimal rooted trees, Discrete Appl. Math., 127, pp. 489-503.

[12] C. Semple and M. Steel (2000), A supertree method for rooted trees, Discrete Appl. Math., 105, pp. 147-158.

[13] C. Semple and M. Steel (2003), Phylogenetics, Oxford University Press.

[14] M. Steel, A. W. M. Dress, and S. Böcker (2000), Simple but fundamental limitations on supertree and consensus tree methods, Syst. Biol., 49, pp. 363-368.

[15] S. Willson (2003), Private communication.

BIOMATHEMATICS RESEARCH CENTRE, DEPARTMENT OF MATHEMATICS AND STATISTICS, UNIVERSITY OF CANTERBURY, CHRISTCHURCH, NEW ZEALAND

E-mail address: pjd62@student.canterbury.ac.nz, c.semple@math.canterbury.ac.nz